

Scalable Computation of Optimized Queries for Sequential Diagnosis

Patrick Rodler and Wolfgang Schmid and Kostyantyn Shchekotykhin

Alpen-Adria Universität, Klagenfurt, 9020 Austria

firstname.lastname@aau.at

Abstract

In many model-based diagnosis applications it is impossible to provide such a set of observations and/or measurements that allow to identify the real cause of a fault. Therefore, diagnosis systems often return many possible candidates, leaving the burden of selecting the correct diagnosis to a user. Sequential diagnosis techniques solve this problem by automatically generating a sequence of queries to some oracle. The answers to these queries provide additional information necessary to gradually restrict the search space by removing diagnosis candidates inconsistent with the answers.

During query computation, existing sequential diagnosis methods often require the generation of many unnecessary query candidates and strongly rely on expensive logical reasoners. We tackle this issue by devising efficient heuristic query search methods. The proposed methods enable for the first time a completely reasoner-free query generation while at the same time guaranteeing optimality conditions, e.g. minimal cardinality or best understandability, of the returned query that existing methods cannot realize. Hence, the performance of this approach is independent of the (complexity of the) diagnosed system. Experiments conducted using real-world problems show that the new approach is highly scalable and outperforms existing methods by orders of magnitude.

Introduction

Model-based diagnosis (MBD) is a principled approach to finding explanations, called diagnoses, for unexpected behavior of observed systems. MBD approaches are applied for, e.g. the diagnosis of hardware (Reiter 1987; de Kleer and Williams 1987; Dressler and Struss 1996; Wotawa 2001) and software (Mateis et al. 2000; Stumptner and Wotawa 1999; Elmishali, Stern, and Kalech 2016), knowledge bases (Friedrich and Shchekotykhin 2005; Kalyanpur et al. 2007), feature models (White et al. 2010), discrete event systems (Pencolé and Cordier 2005; Su, Zanella, and Grastien 2016), user interfaces (Felfernig et al. 2009), etc.

In many cases, however, the observations of a faulty system provide insufficient information for successful fault localization. Therefore, MBD methods might return many possible diagnoses and a user must choose the correct diagnosis manually. Given the complexity of modern systems, selection of the correct fault explanation is a hard task.

Sequential diagnosis methods, e.g. (de Kleer and Williams 1987; Siddiqi and Huang 2011; Pietersma, van

Gemund, and Bos 2005), allow a user to find the correct diagnosis by answering a sequence of queries. Every query is constructed in a way that, given any possible answer of the user, at least one diagnosis is inconsistent with it. Consequently, in every iteration of a sequential method at least one diagnosis is pruned. This approach guarantees identification of the correct diagnosis after a finite number of iterations.

Existing sequential methods rely on strategies for query computation which often lead to the generation of many unnecessary query candidates and thus to a high number of expensive calls to reasoning services. Such an overhead can have a severe impact on the response time of the interactive debugger, i.e. the time between two consecutive queries. Problems of these approaches are (1) the necessity to actually *compute* a query candidate in order to assess its goodness, (2) the very expensive verification whether a candidate is a query, (3) their inability to recognize candidates that are no queries before verification, (4) the possibility of the generation of query duplicates and (5) the strong dependence on the output of used reasoning services regarding the completeness and quality of the set of queries generated.

To tackle these issues, we devise an efficient heuristic query computation algorithm that solves all the mentioned problems of legacy approaches. Our main contributions are:

- We introduce a new, theoretically well-founded and sound method for query generation that works completely *without the use of reasoning services*.
- Our algorithm guarantees optimality of each query w.r.t. (a) the expected number of subsequent queries and (b) required properties, e.g. the number of elements in a query.
- Given a preference criterion, the method can reformulate a query to comprise elements considered simple by a user.
- Evaluation results show the quasi constant execution time of our method for increasing problem size and the significant superiority to existing methods throughout all tests, e.g. for problems with ≥ 15 diagnoses our method always needs less than 1% of the time existing methods require.

To summarize, the proposed query generation method produces a query that is optimized along different dimensions whereas its runtime accounts for only a minor fraction of the reaction time, i.e. the time interval between two queries of a sequential diagnosis method.

\mathcal{K}	$\phi_1 : A \rightarrow B \wedge L$ $\phi_4 : L \rightarrow H$	$\phi_2 : A \rightarrow F$ $\phi_5 : \neg H \rightarrow G \wedge \neg A$	$\phi_3 : B \vee F \rightarrow H$
N	$n_1 : \{A \rightarrow H\}$	$R = \{\text{consistency}\}$	$P, \mathcal{B} = \emptyset$

Table 1: Propositional Logic Example DPI DPI_{ex}

Preliminaries

In this section we review the basics of sequential diagnosis methods. In particular, the approach described in this paper is a variant of a sequential diagnosis technique suggested in (Shchekotykhin et al. 2012). The latter considers the diagnosis problem wrt. interactive KB debugging¹ which, however, can be generalized to various other MBD approaches.

As to the notation, we will write $\mathcal{K} \models X$ for sets of logical formulas \mathcal{K} and X to denote that $\mathcal{K} \models x$ for all $x \in X$. Moreover, U_X and I_X for some collection of sets X refers to the union and the intersection of all sets in X , respectively. In addition, we assume that entailment (\models) is monotonic.

Knowledge Base Debugging. A (*parsimonious*) *KB debugging problem (KDP)* involves finding a maximal solution KB for a diagnosis problem instance given as input. In a *diagnosis problem instance (DPI)* $\langle \mathcal{K}, \mathcal{B}, P, N \rangle_R$ over some logic \mathcal{L} , \mathcal{K} is a KB of potentially faulty formulas, \mathcal{B} a (background) KB of formulas assumed as correct. Further, $R \supseteq \{\text{consistency}\}$ is a set of KB quality requirements a solution KB must meet. P and N are sets of positive and negative test cases, respectively. Each test case is a set (conjunction) of formulas. All elements of \mathcal{K} , \mathcal{B} and of each test case are formulated over the logic \mathcal{L} . A KB \mathcal{K}^* over \mathcal{L} is a *solution KB* for this DPI iff $\mathcal{K}^* \cup \mathcal{B}$ meets all requirements $r \in R$, $\mathcal{K}^* \cup \mathcal{B} \models p$ for all $p \in P$ and $\mathcal{K}^* \cup \mathcal{B} \not\models n$ for all $n \in N$. It is a *maximal solution KB* iff there is no solution KB \mathcal{K}' such that $\mathcal{K}' \cap \mathcal{K} \supset \mathcal{K}^* \cap \mathcal{K}$, i.e. the intersection with the given KB \mathcal{K} must be \subseteq -maximal. A KB \mathcal{K} is said to be *faulty* w.r.t. $\langle \cdot, \mathcal{B}, P, N \rangle_R$ iff $\mathcal{K} \cup \mathcal{B} \cup U_P$ violates some $r \in R$ or entails some $n \in N$. A KB \mathcal{C} is a *conflict set* for $\langle \mathcal{K}, \mathcal{B}, P, N \rangle_R$ iff $\mathcal{C} \subseteq \mathcal{K}$ and \mathcal{C} is faulty w.r.t. $\langle \cdot, \mathcal{B}, P, N \rangle_R$. A conflict set \mathcal{C} is minimal iff there is no conflict set \mathcal{C}' such that $\mathcal{C}' \subset \mathcal{C}$. A set of formulas $\mathcal{D} \subseteq \mathcal{K}$ is called a *diagnosis* for $\langle \mathcal{K}, \mathcal{B}, P, N \rangle_R$ iff $(\mathcal{K} \setminus \mathcal{D}) \cup U_P$ is a solution KB. A diagnosis \mathcal{D} is *minimal* iff there is no diagnosis \mathcal{D}' such that $\mathcal{D}' \subset \mathcal{D}$. The task of a KDP reduces to computing a minimal diagnosis for the given DPI (Rodler 2015, Prop. 3.6). The set of all minimal diagnoses for a DPI DPI is henceforth denoted by $\text{mD}(\text{DPI})$. The relation between minimal conflict sets and minimal diagnoses is as follows (Reiter 1987): A (minimal) diagnosis for a DPI DPI is a (minimal) hitting set of all minimal conflict sets for DPI.

Example: To illustrate these notions, consider the example DPI DPI_{ex} given by Table 1. Denoting KB formulas ϕ_i simply by i , the set of all minimal conflict sets for DPI_{ex} is $\{\{1, 3\}, \{1, 4\}, \{2, 3\}, \{5\}\}$. This holds since each of these sets entails the negative test case $n_1 = \{A \rightarrow H\}$. Further, the set of all minimal diagnoses $\text{mD}(\text{DPI}_{\text{ex}}) = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\} = \{\{1, 2, 5\}, \{1, 3, 5\}, \{3, 4, 5\}\}$ (minimal hitting sets of all minimal conflict sets). All maximal solution KBs are given by $\{3, 4\}$, $\{2, 4\}$ and $\{1, 2\}$. Note that

¹See (Rodler 2015) for a comprehensive treatment.

e.g. both the empty KB \emptyset and $\{X \rightarrow Y\}$ are also solution KBs since both are consistent (meet $r_1 \in R$) and do not entail $n_1 \in N$, albeit not maximal ones. \square

Interactive Knowledge Base Debugging. For one and the same DPI there can be a large number of different (maximal) solution KBs. Each of them has different semantics. Interactive KB debugging aims at restricting the solution space until a single solution (with exactly the desired semantics) is left. Given a DPI $\langle \mathcal{K}, \mathcal{B}, P, N \rangle_R$ as input, an *interactive (dynamic) KB debugging problem (IKDP)* involves finding a maximal solution KB \mathcal{K}^* for a DPI $\langle \mathcal{K}, \mathcal{B}, P', N' \rangle_R$ where $P' \supseteq P$, $N' \supseteq N$ such that \mathcal{K}^* is the only maximal solution KB for this DPI. That is, solving the IKDP means specifying a set $P' \setminus P$ of new positive and a set $N' \setminus N$ of new negative test cases until a single minimal diagnosis is left or, in a relaxed version of IKDP, some minimal diagnosis has a probability higher than some threshold (Shchekotykhin et al. 2012). These new test cases are queries the debugger poses to an interacting user. A query Q is a *t-f-question* if the set of formulas constituted by it must (*t*) or must not (*f*) be true in the intended domain (i.e. entailed by the correct solution KB). An answered query leads to a new DPI, i.e. a positive answer causes the addition of Q to P , a negative one to N .

To be a query, a set of formulas must satisfy two conditions, (1) invalidate at least one minimal diagnosis (search space restriction) and (2) preserve the validity of at least one minimal diagnosis (solution preservation). Formally: A set $Q \neq \emptyset$ of logical formulas over \mathcal{L} is a *query* for the DPI $\langle \mathcal{K}, \mathcal{B}, P, N \rangle_R$ iff $\text{mD}(\langle \mathcal{K}, \mathcal{B}, P, N \rangle_R) \setminus \text{mD}(\langle \mathcal{K}, \mathcal{B}, P \cup \{Q\}, N \rangle_R) \neq \emptyset$ and $\text{mD}(\langle \mathcal{K}, \mathcal{B}, P, N \rangle_R) \setminus \text{mD}(\langle \mathcal{K}, \mathcal{B}, P, N \cup \{Q\} \rangle_R) \neq \emptyset$, i.e. both answers eliminate at least one minimal diagnosis. That is, at least two minimal diagnoses are required to test whether a set of formulas Q is a query. Since the calculation of the entire set $\text{mD}(\langle \mathcal{K}, \mathcal{B}, P, N \rangle_R)$ is generally not tractable within reasonable time due to the complexity of diagnosis computation (Rodler 2015, Sec. 9.4), usually a fixed-cardinality subset \mathbf{D} of it, the *leading diagnoses*, are exploited. In most cases, the latter are the minimal diagnoses of minimal cardinality or maximal probability. Given a DPI $\langle \mathcal{K}, \mathcal{B}, P, N \rangle_R$, we denote by $\mathbf{Q}_{\mathbf{D}, \langle \mathcal{K}, \mathcal{B}, P, N \rangle_R}$ all queries Q for $\langle \mathcal{K}, \mathcal{B}, P, N \rangle_R$ which satisfy $\mathbf{D} \setminus \text{mD}(\langle \mathcal{K}, \mathcal{B}, P \cup \{Q\}, N \rangle_R) \neq \emptyset$ and $\mathbf{D} \setminus \text{mD}(\langle \mathcal{K}, \mathcal{B}, P, N \cup \{Q\} \rangle_R) \neq \emptyset$, i.e. both answers eliminate at least one *leading* diagnosis. For brevity and when the DPI is clear, we will often write only $\mathbf{Q}_{\mathbf{D}}$.

Let $\mathcal{K}_i^* := (\mathcal{K} \setminus \mathcal{D}_i) \cup U_P \cup \mathcal{B}$ for each $\mathcal{D}_i \in \text{mD}(\langle \mathcal{K}, \mathcal{B}, P, N \rangle_R)$, i.e. \mathcal{K}_i^* is the solution KB resulting from the deletion of \mathcal{D}_i along with the background KB. Each query candidate (i.e. set of formulas $Q \neq \emptyset$) partitions the set \mathbf{D} into three parts. These are $\mathbf{D}^+(Q) = \{\mathcal{D}_i \in \mathbf{D} \mid \mathcal{K}_i^* \models Q\}$, i.e. the leading diagnoses consistent only with Q 's positive answer, $\mathbf{D}^-(Q) = \{\mathcal{D}_i \in \mathbf{D} \mid \exists x \in R \cup N : \mathcal{K}_i^* \cup Q \text{ violates } x\}$, i.e. the leading diagnoses consistent only with Q 's negative answer, and $\mathbf{D}^0(Q) = \mathbf{D} \setminus (\mathbf{D}^+(Q) \cup \mathbf{D}^-(Q))$, i.e. the leading diagnoses consistent with both answers. The tuple $\mathfrak{P}(Q) := \langle \mathbf{D}^+(Q), \mathbf{D}^-(Q), \mathbf{D}^0(Q) \rangle$ is called the *q-partition (QP)* of Q for \mathbf{D} iff Q is a query, i.e. iff $Q \in \mathbf{Q}_{\mathbf{D}}$. Thus, not each

partition of \mathbf{D} into three parts is necessarily a QP. Given a QP \mathfrak{P} , we sometimes call its three entries in turn $\mathbf{D}^+(\mathfrak{P})$, $\mathbf{D}^-(\mathfrak{P})$ and $\mathbf{D}^0(\mathfrak{P})$. Q is called *query with (or: for) the QP* \mathfrak{P} iff \mathfrak{P} is the QP of Q . In general (Rodler 2016, Prop. 56), there can be exponentially many (non-equivalent) queries for a single QP. Each query Q is a subset of the common entailments of all KBs in the set $\{\mathcal{K}_i^* \mid \mathcal{D}_i \in \mathbf{D}^+(Q)\}$.

The sets \mathbf{D}^+ and \mathbf{D}^- are a helpful instrument in deciding whether a set of formulas is a query or not. Namely, a non-empty set of formulas X is in $\mathbf{Q}_\mathbf{D}$ iff $\mathbf{D}^+(X) \neq \emptyset$ and $\mathbf{D}^-(X) \neq \emptyset$. That is, for all queries $Q \in \mathbf{Q}_\mathbf{D}$, the first two entries of $\mathfrak{P}(Q)$ must be non-empty. Moreover, $\mathfrak{P}(Q)$ facilitates an estimation of the impact Q 's answers have in terms of the invalidation of minimal diagnoses. And, given (diagnosis) fault probabilities, it enables to gauge the probability of getting a positive or negative answer to a query. Active learning query selection measures $m : Q \mapsto m(Q) \in \mathbb{R}$ (Settles 2012; de Kleer and Williams 1987; Shchekotykhin et al. 2012; Rodler et al. 2013) use exactly these query properties characterized by the QP to assess how favorable a query is.

Example (cont'd): Let the set of leading diagnoses $\mathbf{D} = \mathbf{mD}(\text{DPl}_{\text{ex}}) = \{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3\}$. Then, $Q = \{F \rightarrow H\}$ is a query in $\mathbf{Q}_\mathbf{D}$. To verify this, let us consider its QP $\mathfrak{P}(Q) = \langle \{\mathcal{D}_1\}, \{\mathcal{D}_2, \mathcal{D}_3\}, \emptyset \rangle$. Since both $\mathbf{D}^+(Q)$ and $\mathbf{D}^-(Q)$ are non-empty, Q is in $\mathbf{Q}_\mathbf{D}$. $\mathcal{D}_1 \in \mathbf{D}^+(Q)$ holds as $\mathcal{K}_1^* = (\mathcal{K} \setminus \mathcal{D}_1) \cup \mathcal{B} \cup U_P = (\{1, \dots, 5\} \setminus \{1, 2, 5\}) \cup \emptyset \cup \emptyset = \{3, 4\} = \{B \vee F \rightarrow H, L \rightarrow H\}$ entails Q . On the other hand, e.g. $\mathcal{D}_2 \in \mathbf{D}^-(Q)$ due to the fact that $\mathcal{K}_2^* \cup Q = \{A \rightarrow F, L \rightarrow H, F \rightarrow H\} \models \{A \rightarrow H\} = n_1 \in N$. Hence, Q 's positive answer implies that leading diagnoses in $\mathbf{D}^-(Q) = \{\mathcal{D}_2, \mathcal{D}_3\}$ are invalidated, i.e. are not in $\mathbf{mD}(\langle \mathcal{K}, \mathcal{B}, P \cup \{Q\}, N \rangle_R)$. Likewise, a negative answer causes $\{\mathcal{D}_1\} \cap \mathbf{mD}(\langle \mathcal{K}, \mathcal{B}, P, N \cup \{Q\} \rangle_R) = \emptyset$. \square

Heuristic-Based Query Optimizer (H-QUO)

In this section, we propose a new implementation of the query selection in interactive KB debugging.² Before, we briefly discuss issues that come with existing approaches:

Related Work. To find a query Q with (nearly) optimal value $m(Q)$ w.r.t. a measure m given leading diagnoses \mathbf{D} , existing interactive KB debuggers (Shchekotykhin et al. 2012; Shchekotykhin et al. 2014; Rodler et al. 2013; Rodler 2015) proceed as follows: They (1) use a reasoner *rsnr* to compute for different seeds $\mathbf{D}^+ \subset \mathbf{D}$ a set X of common entailments of all KBs in $\{\mathcal{K}_i^* \mid \mathcal{D}_i \in \mathbf{D}^+\}$ and (2) if $X \neq \emptyset$, then they use *rsnr* to assign all diagnoses in $\mathbf{D} \setminus \mathbf{D}^+$ to their respective set among $\mathbf{D}^+(X)$, $\mathbf{D}^-(X)$ and $\mathbf{D}^0(X)$. (3) If both $\mathbf{D}^+(X)$ and $\mathbf{D}^-(X)$ are non-empty (i.e. X is a query with QP $\mathfrak{P}(X)$) and $m(X)$ is sufficiently good, a \subseteq -minimal subset X' of X is computed using *rsnr* such that $\mathfrak{P}(X') = \mathfrak{P}(X)$.

²The new query computation mechanism H-QUO has already been realized in terms of a Protégé plug-in for sequential/interactive ontology debugging, see <http://isbi.aau.at/ontodebug/>. Protégé is currently the most widely used open-source ontology engineering software in the world, see <http://protege.stanford.edu/>.

In general, steps 1 and 2 are executed $O(2^{|\mathbf{D}|})$ times yielding a worst case exponential number of expensive reasoner calls. (Shchekotykhin et al. 2012) couples this process with a heuristic to direct the search faster towards optimal queries, but partitions suggested by the heuristic might in fact be no QPs which hinders efficient search space pruning. Also, the strong reasoner dependence persists.

As pointed out by (Rodler 2015), all these techniques suffer from the problem that the (number and types of) entailments output by *rsnr*, i.e. the shape of X , have a significant influence on the quality and the number of different calculated queries. (Rodler 2015) shows that objectively worse queries might be found at the cost of neglecting better ones. Further drawbacks are that duplicate QPs might occur in the search (as different seeds can lead to same QP), effective employment of heuristics and pruning is not possible (see above) and that no guarantees whatsoever can be given w.r.t. properties of the minimized query X' . The method H-QUO proposed in this work solves all these issues.

The New Approach. We propose a four-phase optimization of a query (for an in-depth discussion, all algorithms and proofs see (Rodler 2016)). In the first phase (P1), an optimal QP \mathfrak{P} w.r.t. m is computed (FINDQPARTITION). Second (P2), a best query Q for \mathfrak{P} w.r.t. a secondary criterion, e.g. minimum cardinality, is computed (SELECTQUERYFORQPARTITION). Third (P3), Q is enriched by formulas of simple structure, e.g. simple implications of the form $A \rightarrow B$ for Propositional Logic, resulting in $Q' \supseteq Q$ (ENRICHQUERY). Fourth (P4), Q' is minimized and optimized in a way that, roughly, most of the simple formulas are maintained and most of the more complex ones deleted, yielding the final query Q^* (OPTIMIZEQUERY). Note P3 and P4 are optional and require altogether a polynomial number of reasoner calls. P1 and P2 do not require any reasoner. Next, we give the main ideas of the four functions.

P1 – FINDQPARTITION. To overcome the mentioned problems of existing systems, we use the notion of a canonical query, a well-defined subset of \mathcal{K} . Any query $Q \subseteq \mathcal{K}$ in $\mathbf{Q}_\mathbf{D}$ *must* include some formulas in $U_\mathbf{D}$, *need not* include any formulas in $\mathcal{K} \setminus U_\mathbf{D}$, and *must not* include any formulas in $I_\mathbf{D}$ (Rodler 2016, Cor. 19). Moreover, the deletion of any formulas in $\mathcal{K} \setminus U_\mathbf{D}$ from Q does not alter the QP $\mathfrak{P}(Q)$. Hence, we call $\text{Disc}_\mathbf{D} := U_\mathbf{D} \setminus I_\mathbf{D}$ the discrimination formulas (i.e. those that are essential in discriminating between leading diagnoses \mathbf{D}).

Definition 1. Let $\emptyset \subset \mathbf{D}^+ \subset \mathbf{D}$. Then we call $Q_{\text{can}}(\mathbf{D}^+) := (\mathcal{K} \setminus U_{\mathbf{D}^+}) \cap \text{Disc}_\mathbf{D}$ the canonical query (CQ) w.r.t. \mathbf{D}^+ if $Q_{\text{can}}(\mathbf{D}^+) \neq \emptyset$. Otherwise, $Q_{\text{can}}(\mathbf{D}^+)$ is undefined.

In that, $\mathcal{K} \setminus U_{\mathbf{D}^+}$ are exactly the common (explicit) entailments of $\{\mathcal{K} \setminus \mathcal{D}_i \mid \mathcal{D}_i \in \mathbf{D}^+\}$, i.e. the CQ is a minimization of this set under preservation of the QP.

Definition 2. A QP \mathfrak{P}' for which a CQ Q exists with exactly this QP, i.e. $\mathfrak{P}(Q) = \mathfrak{P}'$, is called a canonical QP (CQP).

A restriction – only for now, during P1 – to searching only for CQs, has some nice implications: (1) CQs can be generated by cheap set operations (no reasoner calls), (2) each CQ

is a query in \mathbf{Q}_D for sure, no verification of its QP required, thence no unnecessary query candidates tested, (3) automatic focus on favorable queries (those with empty \mathbf{D}^0), (4) no duplicate QPs as there is a one-to-one relationship between CQs and CQPs.

Example (cont'd): Let \mathbf{D} as before. Then $\text{Disc}_D = U_D \setminus I_D = \{1, 2, 3, 4, 5\} \setminus \{5\} = \{1, 2, 3, 4\}$. Let us consider the seed $\mathbf{D}^+ = \{\mathcal{D}_1\}$. Then the CQ $Q_1 := Q_{\text{can}}(\mathbf{D}^+) = (\{1, \dots, 5\} \setminus \{1, 2, 5\}) \cap \{1, \dots, 4\} = \{3, 4\}$. The associated CQP is $\mathfrak{P}_1 = \langle \{\mathcal{D}_1\}, \{\mathcal{D}_2, \mathcal{D}_3\}, \emptyset \rangle$. Note $\mathcal{D} \in \mathbf{D}^+(\mathfrak{P}_1)$ ($\mathcal{D} \in \mathbf{D}^-(\mathfrak{P}_1)$) for some $\mathcal{D} \in \mathbf{D}$ iff $\mathcal{K} \setminus \mathcal{D} \supseteq \{3, 4\} = Q_1$. E.g. $\mathcal{D}_3 \in \mathbf{D}_1^-$ since $(\mathcal{K} \setminus \mathcal{D}_3) = \{1, 2\} \not\supseteq \{3, 4\} = Q_1$. That is, reasoning is traded for set operations and comparisons.

The seed $\mathbf{D}^+ = \{\mathcal{D}_1, \mathcal{D}_3\}$ yields $Q_2 := Q_{\text{can}}(\mathbf{D}^+) = (\{1, \dots, 5\} \setminus \{1, 2, 5\}) \cap \{1, \dots, 4\} = \emptyset$, i.e. there is no CQ w.r.t. \mathbf{D}^+ and the partition $\langle \{\mathcal{D}_1, \mathcal{D}_3\}, \{\mathcal{D}_2\}, \emptyset \rangle$ with the seed as first entry is no CQP (and also no QP). \square

Since the evaluation of $m(Q)$ for a query Q requires the QP $\mathfrak{P}(Q)$, we propose a (heuristic) depth-first, local best-first (i.e. chooses only among best direct successors at each step) backtracking search procedure that abstracts from queries and focuses only on QPs, in particular only on CQPs (and thence on CQs). The optimal query for the found QP is determined separately in the next phase P2. Note this separation is not possible without the notion of a CQ.

A search problem (Russell and Norvig 2010) is defined by the *initial state*, a *successor function* enumerating all direct neighbor states of a state, the *step costs* from a state to a successor state, some *heuristics* to estimate the remaining effort towards a goal state, and the *goal test* to determine if a given state is a goal state or not. We define the initial state as $\langle \emptyset, \mathbf{D}, \emptyset \rangle$ (note this is not a QP). The idea is to transfer diagnoses step-by-step from \mathbf{D}^- to \mathbf{D}^+ to construct all CQPs systematically. The step costs are irrelevant, only the found QP as such counts, not the way to get there. Heuristics can be formulated based on qualitative requirements on optimal queries which can be derived from the real-valued function m and used for search space pruning (as shown for numerous measures in (Rodler 2016)). A QP is considered a goal if it optimizes m up to some threshold t (cf. (Shchekotykhin et al. 2012)). In order to characterize a suitable successor function, we define a direct neighbor of a QP by means of:

Definition 3. Let $\mathfrak{P}_i := \langle \mathbf{D}_i^+, \mathbf{D}_i^-, \emptyset \rangle$, $\mathfrak{P}_j := \langle \mathbf{D}_j^+, \mathbf{D}_j^-, \emptyset \rangle$ be partitions of \mathbf{D} . Then, $\mathfrak{P}_i \mapsto \mathfrak{P}_j$ is a minimal \mathbf{D}^+ -transformation from \mathfrak{P}_i to \mathfrak{P}_j iff \mathfrak{P}_j is a CQP, $\mathbf{D}_i^+ \subset \mathbf{D}_j^+$ and there is no CQP $\langle \mathbf{D}_k^+, \mathbf{D}_k^-, \emptyset \rangle$ with $\mathbf{D}_i^+ \subset \mathbf{D}_k^+ \subset \mathbf{D}_j^+$.

From now on, we call a CQP \mathfrak{P}' a successor of a partition \mathfrak{P} iff \mathfrak{P}' results from \mathfrak{P} by a minimal \mathbf{D}^+ -transformation. So, all successors of the initial state have the form $\langle \{\mathcal{D}_i\}, \mathbf{D} \setminus \{\mathcal{D}_i\}, \emptyset \rangle$ (Rodler 2016, Cor. 20). To specify successors of an intermediate CQP \mathfrak{P}_k , we draw on traits:

Definition 4. Let $\mathfrak{P}_k = \langle \mathbf{D}_k^+, \mathbf{D}_k^-, \emptyset \rangle$ be a CQP and $\mathcal{D}_i \in \mathbf{D}_k^-$. Then the trait $\mathcal{D}_i^{(k)}$ of \mathcal{D}_i is defined as $\mathcal{D}_i \setminus U_{\mathbf{D}_k^+}$.

The relation \sim_k associating two diagnoses in \mathbf{D}_k^- iff their trait is equal is an equivalence relation. Now, all successors of \mathfrak{P}_k have the form $\langle \mathbf{D}_k^+ \cup E, \mathbf{D}_k^- \setminus E, \emptyset \rangle$ where

E is an equivalence class with a \subseteq -minimal trait among all equivalence classes $\{E_1, \dots, E_s\}$ w.r.t. \sim_k and $s \geq 2$. And, there are no other successors of \mathfrak{P}_k . Usage of this successor function makes the search for CQPs complete. For the number n_{CQP} of CQPs it holds $n_{\text{CQP}} = |\{U_{\mathbf{D}^+} \mid \emptyset \subset \mathbf{D}^+ \subset \mathbf{D}, U_{\mathbf{D}^+} \neq U_{\mathbf{D}}\}| \geq |\mathbf{D}|$ (Rodler 2016, Cor. 22). It is an open question whether QPs of the form $\langle \mathbf{D}^+, \mathbf{D}^-, \emptyset \rangle$ exist which are no CQPs, but (Rodler 2016, Sec. 3.4.2) gives theoretical and empirical evidence that, if there are such, then they should be quite rare. Moreover, our evaluation manifests that there are large numbers of CQPs in relation to $|\mathbf{D}|$ which grants the detection of optimal QPs w.r.t. all measures m discussed in the KB debugging literature (Shchekotykhin et al. 2012; Rodler et al. 2013).

Example (cont'd): Reconsider the CQP \mathfrak{P}_1 . The traits are $\mathcal{D}_2^{(1)} = \{1, 3, 5\} \setminus \{1, 2, 5\} = \{3\}$ and $\mathcal{D}_3^{(1)} = \{3, 4\}$, representing two equivalence classes w.r.t. \sim_1 . There is only one class with \subseteq -minimal trait, i.e. $\{\mathcal{D}_2\}$. Hence, there is a single successor CQP $\mathfrak{P}_2 = \langle \{\mathcal{D}_1, \mathcal{D}_2\}, \{\mathcal{D}_3\}, \emptyset \rangle$ of \mathfrak{P}_1 . Recall, we argued that $\langle \{\mathcal{D}_1, \mathcal{D}_3\}, \{\mathcal{D}_2\}, \emptyset \rangle$ is indeed no CQP. \square

P2 – SELECTQUERYFORQPARTITION. The QP \mathfrak{P}_k returned by P1 is already optimal w.r.t. the (e.g. information theoretic) measure m . Such measures aim at the minimization of subsequent queries until the correct diagnosis is found. Also, we have a CQ Q_k for \mathfrak{P}_k . However, usually a minimal requirement is the \subseteq -minimality of a query to reduce the effort for the user. To this end, let $\text{Tr}(\mathfrak{P}_k)$ denote the set of all \subseteq -minimal traits w.r.t. \sim_k . Then, $Q \subseteq \text{Disc}_D$ is a \subseteq -minimal query with QP \mathfrak{P}_k iff $Q = H$ for some minimal hitting set of $\text{Tr}(\mathfrak{P}_k)$ (Rodler 2016, Cor. 24).

Hence, all \subseteq -minimal reductions of CQ under preservation of the (already fixed and optimal) QP \mathfrak{P}_k can be computed e.g. using the classical HS-TREE (Reiter 1987). However, there is a crucial difference to standard application scenarios of HS-TREE, namely the fact that all sets to label the tree nodes (i.e. the \subseteq -minimal traits) are readily available (without further computations). Consequently, the construction of the tree runs swiftly, as our evaluation will confirm (note also, in principle we only require a single minimal hitting set). Moreover, HS-TREE can be used as uniform-cost search (cf. e.g. (Rodler 2015, Chap. 4)), incorporating various quality criteria *crit* such as minimum cardinality or best comprehensibility (by means of given fault information (Rodler 2016, Sec. 3.5)) of the output query. No existing approach can realize such optimizations. So far, P1 and P2 have computed – without any call to a reasoner – an optimal query Q w.r.t. m (# of queries) and *crit* (effort per query) which can be directly shown to the user. Optionally, Q can be further enhanced in phases P3 and P4.

Example (cont'd): Recall the CQP \mathfrak{P}_1 . Then, $\text{Tr}(\mathfrak{P}_1) = \{\{3\}\}$, i.e. there is a single minimum cardinality query $\{3\}$ for \mathfrak{P}_1 , a proper subset of the CQ $\{3, 4\}$ for \mathfrak{P}_1 . Considering the CQP $\mathfrak{P}_3 := \langle \{\mathcal{D}_2\}, \{\mathcal{D}_1, \mathcal{D}_3\}, \emptyset \rangle$, we have $\text{Tr}(\mathfrak{P}_3) = \{\{2\}, \{4\}\}$ and thus a single query $\{2, 4\}$ of minimal size which is equal to the CQ for \mathfrak{P}_3 . \square

P3 – ENRICHQUERY. Sometimes a user might find it hard to classify formulas occurring explicitly in the KB as true

or false statements in the intended domain, e.g. users are often convinced of the correctness of formulas they specified themselves. In such a case it might be easier, less error-prone and more convenient for users to be presented with a query including *simple* formulas not in the KB. In case of e.g. OWL or Description Logics, such formulas might be subsumption or class assertion axioms (Baader et al. 2007).

Therefore, ENRICHQUERY realizes the expansion of the given query $Q \subseteq \mathcal{K}$ returned by P2 by (finitely many) additional formulas Q_{impl} . At this, we postulate that (1) $Q_{impl} \cap \mathcal{K} \cup \mathcal{B} \cup U_P = \emptyset$ (only implicit entailments), (2) $S \models Q_{impl}$ where S is some solution KB for the given DPI and $Q \subseteq S \subseteq \mathcal{K} \cup \mathcal{B} \cup U_P$ (entailed by a consistent, non-faulty set of formulas) and (3) no $\phi_i \in Q_{impl}$ is an entailment of $S \setminus Q$ (logical dependence on Q , no irrelevant formulas). Further on, we require that (4) the query expansion does not alter the (already fixed and optimal) q-partition.

We define the expansion of Q resulting in $Q' \leftarrow Q \cup Q_{impl}$ as $Q_{impl} := [F^{(+Q)} \setminus F^{(-Q)}] \setminus Q$ where

$$F^{(+Q)} := Ent_T[(\mathcal{K} \setminus U_D) \cup \mathcal{B} \cup U_P \cup Q]$$

$$F^{(-Q)} := Ent_T[(\mathcal{K} \setminus U_D) \cup \mathcal{B} \cup U_P]$$

and $Ent_T()$ is the function computing entailments of predefined formula types T implemented by a reasoner. Q_{impl} specified this way satisfies all postulated requirements (1) – (3) (Rodler 2016, Prop. 57) and Q' meets requirement (4) (Rodler 2016, Prop. 58). Note that only two reasoner calls are required by the query enrichment.

P4 – OPTIMIZEQUERY. The enriched query Q' returned by P3 is optimized in this phase. The plausible assumption now is that none of the simple formulas in Q_{impl} is harder to understand than any formula in Q . And, the higher the fault probability of a formula is, the harder it is to understand for the user. The objective of OPTIMIZEQUERY is (1) the q-partition-preserving minimization of Q' to obtain a \subseteq -minimal subset Q^* of Q' (low effort for user), (2) $Q^* \cap Q = \emptyset$ (maintain only simple formulas), if such a Q^* exists, and otherwise (3) if formula fault probabilities $p(\phi)$ for $\phi \in \mathcal{K}$ are given, then Q^* is required to be the one query among all \subseteq -minimal subsets of Q' that minimizes the maximum probability $p(\phi)$ over all $\phi \in Q$ occurring in it (make hardest formula as simple as possible).

Requirement (1) can be easily achieved by applying to Q' the q-partition-preserving divide-and-conquer query minimizer MINQ (a modification of QUICKXPLAIN (Junker 2004)) used in e.g. (Shchekotykhin et al. 2012; Rodler et al. 2013) and described comprehensively in (Rodler 2015, Sec. 8.3). However, to additionally meet conditions (2) and (3), we modify the input to MINQ accordingly. In particular, we use the following sorting $[Q_{impl}, asc_p(Q)]$ of formulas in Q' where $[X, Y]$ denotes a list containing first (i.e. left-most) all elements of the collection X and then all elements of the collection Y , and $asc_{crit}(X)$ refers to the list of elements in the collection X sorted ascending based on *crit*.

Then, the application of MINQ to the sorted enriched query $[Q_{impl}, asc_p(Q)]$ returns a query Q^* compliant with requirements (1) – (3) (Rodler 2016, Cor. 25).

Evaluation

We conducted two experiments Exp1 (scalability) and Exp2 (comparison with standard approach) using the faulty real-world KBs University (U), Transportation (T) and Economy (E) from an OWL ontology dataset (Kalyanpur et al. 2007). We chose these three KBs since their number of minimal diagnoses (90, 1782, 864) (Shchekotykhin et al. 2012), is large enough for our tests, and since there are sound and complete OWL reasoners like Hermit (Shearer, Motik, and Horrocks 2008) which we used in our tests for consistency checks and the computation of classification and realization entailments (Baader et al. 2007). In both experiments, we used the direct diagnosis generation technique INV-HS-TREE presented in (Shchekotykhin et al. 2014) to generate for each iteration a set of minimal diagnoses \mathbf{D} where $|\mathbf{D}| \in \{10, 20, \dots, 90\}$ in Exp1 and $|\mathbf{D}| \in \{5, 10, 15, 20\}$ in Exp2. To always obtain a different set \mathbf{D} , we incorporated a random re-sorting of the KB-formulas before each call of INV-HS-TREE, cf. (Shchekotykhin et al. 2014). Each test run involved 5 iterations for fixed KB and $|\mathbf{D}|$. In each iteration of (a) Exp1, we used our new approach H-QUO with (heuristics for) one of the query selection strategies split-in-half (SPL) or entropy (ENT) (Shchekotykhin et al. 2012) to generate an optimized query per iteration (i.e. performing all four phases P1 – P4), (b) Exp2, we used H-QUO just as in Exp1 and additionally applied the standard method (see Related Work), Std for short, to search different random fractions (full, $\frac{2}{3}$, $\frac{1}{3}$) of the QP search space. Note, in each iteration of Exp2, H-QUO and Std were carried out with exactly the same settings.

Results of Exp1. Fig. 1 (left side) shows the results for all runs where no timeout (execution time > 1 hour) occurred in any iteration. For all runs, if a timeout occurred, it was due to diagnosis computation time (see blue lines). On the contrary, the query computation time – which is almost equal for SPL, ENT (see bars) – is give or take independent of $|\mathbf{D}|$, as indicated by the gray continuous lines. These give the average over $\{SPL, ENT\}$ of query computation time needed for P1+2 (FINDQPARTITION and SELECTQUERYFORQPARTITION) and P3+4 (ENRICHQUERY and OPTIMIZEQUERY). We can see that the former is always negligible (never more than 0.03% of overall debugger reaction time, see dotted line) and the latter requires never more than 31.3 sec (case E70), even though considering QP search space sizes of up to $O(2^{80})$. Of the time consumed by P3+4, reasoning time (polynomial number of calls to a reasoner) never amounted to less than 92%, for the most costly cases (KBs T and E) even more than 99% (dashed line at the top). So, the mentioned 31.3 sec include exactly 31.07 sec reasoning time. Note, since the execution of P1+2 happens instantaneously (always less than 0.01 sec) thanks to the reasoner avoidance, the line for P3+4 maps at the same time the overall costs of H-QUO. For increasing $|\mathbf{D}|$, these have an asymptotically negligible influence on the reaction time of a debugger (decreasing dashed line). Hence, with H-QUO, query computation is not the bottleneck anymore, as opposed to the usage of standard methods (see Exp2). In other words, if a set of leading diagnoses \mathbf{D} of a given cardinality is computable in reasonable time, so is a query (which is already optimized along various dimensions!).

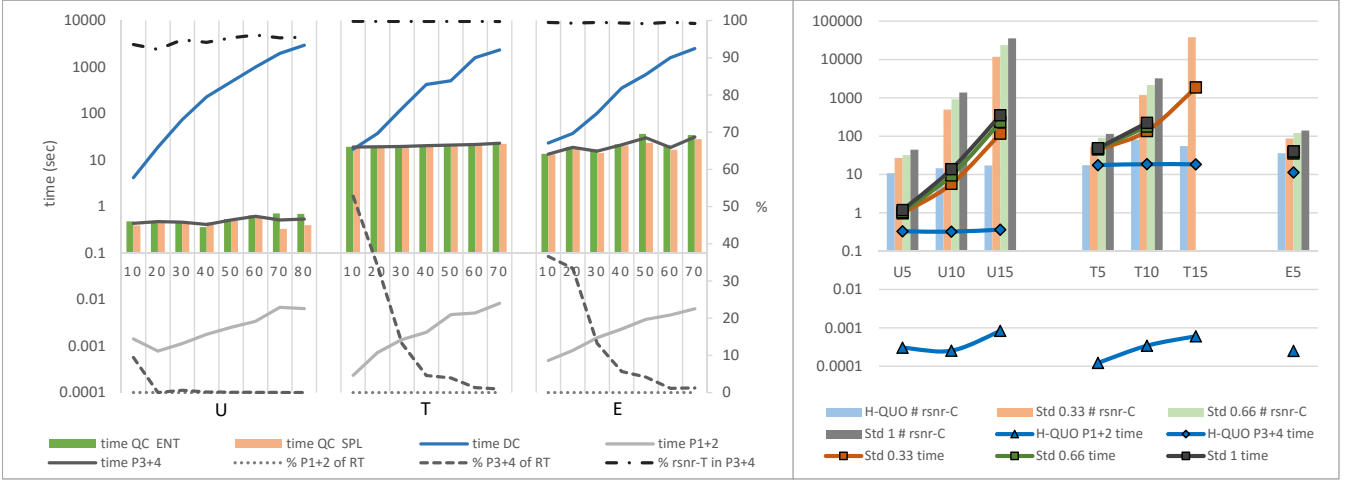


Figure 1: Experiment results for Exp1 (left) and Exp2 (right). All values are 5 trial averages. The x-axes enumerate (KB,|D|)-configurations. **(Left):** H-QUO only. Shows times (left *logarithmic* y-axis, continuous lines and bars) for query computation (QC) using $m \in \{\text{ENT}, \text{SPL}\}$, for phases P1+2 and P3+4 (averaged over ENT, SPL) separately and for diagnosis computation (DC), and % (right y-axis, dashed lines) of debugger reaction time (RT), i.e. time between two queries, spent for P1+2 and P3+4 and % of time used for reasoning (rsnr-T) during P3+4. **(Right):** H-QUO vs. standard methods (Std). Times and # of reasoner calls (rsnr-C) shown by lines and bars, respectively. The scale on the y-axis is logarithmic. In “Std i ” the i denotes the fraction of the QP search space considered. Where no bars or lines are shown, a 1-hour-timeout occurred.

Further, over all runs in Exp1, the average (1) CQ size was 5.9, (2) returned query size was 3.0 (ENT) and 2.7 (SPL) formulas, (3) number of expanded and generated CQPs during P1 was 9.5 and 85.5, (4) number of CQP successors was 10, (5) trait size was 1.6 (very small variance over different (KB,|D|)-runs), (6) number of expanded and generated HS nodes during P2 was 3.6 and 5.1 (small variance over different (KB,|D|)-runs), (7) size reduction of CQ achieved by P2 was 43%, (8) query size increase due to enrichment in P3 was independent of |D|, but strongly dependent on the KB (number of implicit entailments) and amounted to 14% (U), 2250% (T) and 78200% (factor 783) (E), (9) query size reduction through P4 was 13% (U), 96% (T) and 99.8% (E).

Results of Exp2. Fig. 1 (right side) depicts the results for all runs where no timeout (execution time > 1 hour) occurred in any iteration. Unlike in Exp1, here timeouts occurred *only* for Std and due to query computation time (diagnosis computation took never longer than 4 min per iteration). Like in the scalability experiment (Exp1), H-QUO manifests constant behavior despite growing |D|. We point out that for |D| = 10, which is already larger than the leading diagnoses size commonly used (Shchekotykhin et al. 2012; Rodler et al. 2013), H-QUO is already one order of magnitude (logarithmic, base 10 axis) better than Std 0.33 (see blue diamond and orange lines), even though Std considers only one third of the QP search space in this setting (hence suboptimal queries are possible). Note that the time for P3+4 can again be seen as overall time for H-QUO (see negligible values along blue triangle P1+2 line). For |D| = 15, the time improvement already exceeds two orders of magnitude, i.e. 99%. Whereas no reasoner is required for P1+2, P3+4 exhibits more than one order of magnitude fewer reasoner calls than Std in all runs where $n \geq 10$. The

similarity between orange, green and gray lines further reveals that limiting the search space to a constant fraction has almost no influence on the scalability of Std, as the QP search space grows exponentially. Except for the T15 case (where Std 0.33 falls still slightly short of the timeout), if one Std setting fails, all do so. What is more, for the KB E, where reasoning is more costly, none of the Std setups with |D| > 5 could finish within an hour. In conjunction with the constant execution time of H-QUO throughout the scalability tests (Exp1), this means that H-QUO requires only a fraction of less than 1% of the execution time of Std for all cases where |D| ≥ 15 .

Conclusions

In this work we have presented a new query generation and optimization approach. Given a number of diagnoses, it allows for the computation of a query that has (a) optimal discrimination properties (minimal expected number of subsequent queries to be answered) and (b) minimal query cardinality or best comprehensibility (minimal effort for the user per query). Importantly, this can be accomplished without any usage of reasoning services. Optionally, however, the latter can be utilized to further enhance an optimized query by replacing potentially complex query elements by simpler surrogates without harming the query’s optimal properties. An extensive evaluation using real-world problems testifies the perfect scalability of the approach – quasi constant time performance for increasing problem size and numbers of diagnoses – and shows that it drastically outperforms existing methods. In particular, our algorithm outputs an optimal query in less than 0.01 sec for all tested problems and search space sizes of up to $O(2^{80})$.

References

- [Baader et al. 2007] Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2007. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.
- [de Kleer and Williams 1987] de Kleer, J., and Williams, B. C. 1987. Diagnosing multiple faults. *Artificial Intelligence* 32(1):97–130.
- [Dressler and Struss 1996] Dressler, O., and Struss, P. 1996. The consistency-based approach to automated diagnosis of devices. *Principles of Knowledge Representation* 269–314.
- [Elmishali, Stern, and Kalech 2016] Elmishali, A.; Stern, R.; and Kalech, M. 2016. Data-augmented software diagnosis. In Schuurmans, D., and Wellman, M. P., eds., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, 4003–4009. AAAI Press.
- [Felfernig et al. 2009] Felfernig, A.; Friedrich, G.; Isak, K.; Shchekotykhin, K. M.; Teppan, E.; and Jannach, D. 2009. Automated debugging of recommender user interface descriptions. *Applied Intelligence* 31(1):1–14.
- [Friedrich and Shchekotykhin 2005] Friedrich, G., and Shchekotykhin, K. 2005. A General Diagnosis Method for Ontologies. In Gil, Y.; Motta, E.; Benjamins, R.; and Musen, M., eds., *Proceedings of the 4th International Semantic Web Conference (ISWC 2005)*, 232–246. Springer.
- [Junker 2004] Junker, U. 2004. QUICKXPLAIN: Preferred Explanations and Relaxations for Over-Constrained Problems. In McGuinness, D. L., and Ferguson, G., eds., *Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence*, volume 3, 167–172. AAAI Press / The MIT Press.
- [Kalyanpur et al. 2007] Kalyanpur, A.; Parsia, B.; Horridge, M.; and Sirin, E. 2007. Finding all Justifications of OWL DL Entailments. In Aberer, K.; Choi, K.-S.; Noy, N. F.; Allemang, D.; Lee, K.-I.; Nixon, L. J. B.; Golbeck, J.; Mika, P.; Maynard, D.; Mizoguchi, R.; Schreiber, G.; and Cudré-Mauroux, P., eds., *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007*, volume 4825 of *LNCS*, 267–280. Berlin, Heidelberg: Springer Verlag.
- [Mateis et al. 2000] Mateis, C.; Stumptner, M.; Wieland, D.; and Wotawa, F. 2000. Model-Based Debugging of Java Programs. In *AADEBUG'00*.
- [Pencolé and Cordier 2005] Pencolé, Y., and Cordier, M.-O. 2005. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence* 164(1):121–170.
- [Pietersma, van Gemund, and Bos 2005] Pietersma, J.; van Gemund, A. J.; and Bos, A. 2005. A model-based approach to sequential fault diagnosis. In *IEEE Autotestcon, 2005.*, 621–627. IEEE.
- [Reiter 1987] Reiter, R. 1987. A Theory of Diagnosis from First Principles. *Artificial Intelligence* 32(1):57–95.
- [Rodler et al. 2013] Rodler, P.; Shchekotykhin, K.; Fleiss, P.; and Friedrich, G. 2013. RIO: Minimizing User Interaction in Ontology Debugging. In Faber, W., and Lembo, D., eds., *Web Reasoning and Rule Systems*, volume 7994 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg. 153–167.
- [Rodler 2015] Rodler, P. 2015. *Interactive Debugging of Knowledge Bases*. Ph.D. Dissertation, Alpen-Adria Universität Klagenfurt. <http://arxiv.org/pdf/1605.05950v1.pdf>.
- [Rodler 2016] Rodler, P. 2016. Towards Better Response Times and Higher-Quality Queries in Interactive Knowledge Base Debugging. Technical report, Alpen-Adria Universität Klagenfurt. <http://arxiv.org/pdf/1609.02584v1.pdf>.
- [Russell and Norvig 2010] Russell, S. J., and Norvig, P. 2010. *Artificial Intelligence: A Modern Approach*. Pearson Education, 3rd edition.
- [Settles 2012] Settles, B. 2012. *Active Learning*. Morgan and Claypool Publishers.
- [Shchekotykhin et al. 2012] Shchekotykhin, K.; Friedrich, G.; Fleiss, P.; and Rodler, P. 2012. Interactive Ontology Debugging: Two Query Strategies for Efficient Fault Localization. *Web Semantics: Science, Services and Agents on the World Wide Web* 12-13:88–103.
- [Shchekotykhin et al. 2014] Shchekotykhin, K.; Friedrich, G.; Rodler, P.; and Fleiss, P. 2014. Sequential diagnosis of high cardinality faults in knowledge-bases by direct diagnosis generation. In *Proceedings of the 21st European Conference on Artificial Intelligence (ECAI 2014)*. IOS Press.
- [Shearer, Motik, and Horrocks 2008] Shearer, R.; Motik, B.; and Horrocks, I. 2008. HermiT : A Highly-Efficient OWL Reasoner. In *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008 EU)*.
- [Siddiqi and Huang 2011] Siddiqi, S., and Huang, J. 2011. Sequential diagnosis by abstraction. *Journal of Artificial Intelligence Research* 41:329–365.
- [Stumptner and Wotawa 1999] Stumptner, M., and Wotawa, F. 1999. Debugging functional programs. In *IJCAI'99*, 1074–1079.
- [Su, Zanella, and Grastien 2016] Su, X.; Zanella, M.; and Grastien, A. 2016. Diagnosability of discrete-event systems with uncertain observations. In Kambhampati, S., ed., *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 1265–1271. IJCAI/AAAI Press.
- [White et al. 2010] White, J.; Benavides, D.; Schmidt, D. C.; Trinidad, P.; Dougherty, B.; and Cortés, A. R. 2010. Automated diagnosis of feature model configurations. *Journal of Systems and Software* 83(7):1094–1107.
- [Wotawa 2001] Wotawa, F. 2001. Debugging Hardware Designs Using a Value-Based Model. *Applied Intelligence* 16(1):71–92.